Compressive Sensing

Dr.-Ing. Ali Bereyhi

Friedrich-Alexander University

ali.bereyhi@fau.de

Summer 2022

Why Considering Random Objects in Compressive Sensing?

There are questions which are hard to answer deterministically

- *Matrices which satisfy the restricted isometry property*
- *Minimum compression rate for exact sparse recovery*
- *Optimal scheme for noisy sparse recovery*

Random models help in such cases

They give a framework in which

Optimality and validity check are possible

We see some examples in this part

Random Sensing

Random Sensing Matrices

For recovery guarantee we need a matrix which satisfies

restricted isometry property

Theoretically, we can do it by

$$M \geq Cs$$

rows for some constant C

How can we make such a matrix?

This is not easy task to make such a matrix deterministically

Random Sensing Matrices

How do we do it by random matrices?

Let matrix $\mathbf{A}$ *be generated randomly as follows:*

- *Each entry is independent and identically distributed*
- *Entries are Gaussian with variance* $1/M$ *and mean zero*

We call this matrix a Gaussian random matrix

Random Sensing Matrices

This means

$$\mathbf{A} = \begin{bmatrix} A_{11} & A_{12} & \dots & A_{1N} \\ \vdots & & \dots & \vdots \\ A_{M1} & A_{M2} & \dots & A_{MN} \end{bmatrix}$$

A_{mn} are independent and

$$A_{mn} \sim \mathcal{N}(0, 1/M)$$

for all m and n

Random Sensing Matrices

Since $\mathbf{A}$ is random, $\delta_s(\mathbf{A})$ is also random

$$\boxed{\textit{By changing the realization of } \mathbf{A}, \ \delta_s(\mathbf{A}) \textit{ also changes}}$$

We could can calculate the following probability

$$P_s(\delta^\star) = \Pr\{\delta_s(\mathbf{A}) \leq \delta^\star\}$$

If this probability is close to one, we could conclude that

$$\boxed{\textit{With high probability a realization of } \mathbf{A} \textit{ satisfies RIP}}$$

Random Sensing Matrices

For example, assume that

$$P_s\left(\delta^\star\right) = 0.99$$

This means that

From every 100 realizations of $\mathbf{A}$, *99 of them satisfy RIP*

As a result, if we generate $\mathbf{A}$ *randomly as said*

with high probability, the system works fine

Random Sensing Matrices

Although checking RIP for a deterministic matrix is hard,

calculating $P_s(\delta^\star)$ for the random $\mathbf{A}$ is possible

This is why we go for random matrices

We generate the matrix randomly

and with high probability sampling and recovery are successful

Random Sensing Matrices

RIP of Gaussian Matrices

Let **A** *be a Gaussian random matrix. Then, there exists a fixed constant C, such that*

$$P_s(\delta) \geq 1 - \epsilon$$

if the number of rows satisfies

$$M \geq \frac{C}{\delta^2}\left(s + s\log\frac{N}{s} + \log\frac{2}{\epsilon}\right)$$

Random Sensing Matrices

What does this result say?

If we want to have $P_s(\delta) \geq 1 - \epsilon$, we need to set

$$M \geq \frac{C}{\delta^2} \left(s + s \log \frac{N}{s} + \log \frac{2}{\epsilon} \right)$$

in our random matrix. Then RIP is high probably satisfied

Let's try to see how this work in practice

Random Sensing Matrices

Assume we have the following scenario:

- *An MRI with N pixels whose only $s = 0.01N$ are non-zero*
- *Basis pursuit is used which recovers image perfectly, if*

$$\delta_{2s}(\mathbf{A}) \leq \frac{1}{3}$$

We generate the matrix randomly and set

$$P_{2s}\left(\frac{1}{6}\right) \geq 0.999$$

So that we make sure that with high probably we recover perfectly

Random Sensing Matrices

Assume we have the following scenario:

- *An MRI with N pixels whose only $s = 0.01N$ are non-zero*
- *Basis pursuit is used which recovers image perfectly, if*

$$\delta_{2s}\left(\mathbf{A}\right) \leq \frac{1}{3}$$

If we could have used optimal recovery, we would have needed

$$M \geq 2s = 0.02N$$

samples!

Random Sensing Matrices

Assume we have the following scenario:

- *An MRI with N pixels whose only $s = 0.01N$ are non-zero*
- *Basis pursuit is used which recovers image perfectly, if*

$$\delta_{2s}(\mathbf{A}) \leq \frac{1}{3}$$

In this system, we need to have

$$M \geq 9C\left(s + 4.6s + 7.6\right) \approx 50Cs + 68.5C$$

samples

Random Sensing Matrices

Assume we have the following scenario:

- *An MRI with N pixels whose only $s = 0.01N$ are non-zero*
- *Basis pursuit is used which recovers image perfectly, if*

$$\delta_{2s}(\mathbf{A}) \leq \frac{1}{3}$$

Now assume that we know $C \leq 0.08$, then we could say

$$M \geq 4s + 6 = 0.04N + 6$$

Random Sensing Matrices

Assume we have the following scenario:

- *An MRI with N pixels whose only $s = 0.01N$ are non-zero*
- *Basis pursuit is used which recovers image perfectly, if*

$$\delta_{2s}\left(\mathbf{A}\right) \leq \frac{1}{3}$$

Usually,an MRI has around $N = 10^5$ pixels which means

$$0.04N + 6 \approx 0.04N$$

Only double measurements!

Random Sensing Matrices

So what does that mean?

We could simply

- *sample the signal randomly*
- *use the tractable* algorithm *of basis pursuit*

We will then have only 0.1% *chance of failure*

by only taking double more samples compared to minimum

Random Sensing Matrices

What if we want to have even less chance of failure?

Say we set $\epsilon = 10^{-5}$*; then we need*

$$M \geq 50Cs + 110C \approx 0.04N + 9$$

We will then have only 0.001% *chance of failure*

by adding only 3 extra samples!

which makes totally no impact in a real system

Random Sensing Matrices

A valid approximation for Gaussian matrices is that

The chance of failure ϵ is almost zero, if we set

$$M > 2s \log\left(\frac{N}{s}\right) = M_1$$

We now compare it with the optimal bound

$$M > 2s = M_0$$

Random Sensing Matrices

As the fraction of zero entries increase, i.e., N/s grows large,

$$\Delta M = M_1 - M_0 = 2s\left(\log\frac{N}{s} - 1\right)$$

grows large

Attention!

This approximation is accurate only when $N \gg s$

Random Sensing Matrices : Moral of Story

A valid approximation for Gaussian matrices is that

The chance of failure ϵ is almost zero, if we set

$$M > 2s \log\left(\frac{N}{s}\right)$$

This means that

With more sparsity, we get much sub-optimal

However, this is always a matter of some small factor

Random Sensing Matrices: Few Notes

There are few notes to pay attention to

- *There are various random matrices with such properties*
- *The bound on the number of samples is different for each*
- *The best choice depends on various factors, e.g.,*
  - *Signal dimension*
  - *Sparsity*
  - *Recovery algorithm*
- *With large N and small ϵ, all bounds are approximately*

$$M > 2s \log\left(\frac{N}{s}\right)$$

Stochastic Compressive Sensing

Stochastic Sparse Signals

Sparse signals are often derived from a random process

In MRI, the pixels are

reflections from a body organ

and so randomly change from a person to another

How we could take into account this randomness?

We should consider a stochastic model for a sparse signal

Stochastic Compressive Sensing

Section 1: Stochastic Model for Sparse Signals

Stochastic Model for Sparse Signals

An stochastic signal in a finite dimension is a random vector

$$\boxed{\textit{Its distribution describes our belief on values signal could take}}$$

Example: *Consider the following stochastic signal*

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_N \end{bmatrix} : \ x_n \text{ are independent uniform Bernoullis}$$

This means that we believe that

$\boldsymbol{x}$ is binary and there are almost same zeors and ones

Stochastic Model for Sparse Signals

What is our belief about a sparse signal?

It has few non-zeros and the rest of entries are zeros

How can we model it stochastically?

A conventional model is

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_N \end{bmatrix} : x_n = b_n u_n$$

for independent and identically distributed b_n *and* u_n

Stochastic Model for Sparse Signals

What is our belief about a sparse signal?

It has few non-zeros and the rest of entries are zeros

How can we model it stochastically?

Each of the components of $x_n = b_n u_n$ are as follows

- *b_n is a Bernoulli random variable*

$$\Pr\{b_n = 1\} = 1 - \Pr\{b_n = 0\} = \alpha$$

- *u_n is a random variable with $\Pr\{u_n = 0\} = 0$*

Good example: Gaussian random variable

Stochastic Model for Sparse Signals

Why is it a good model for sparse signals?

Remember typicality from information theory

$$\frac{1}{N}\left\{\# \textit{ of zeros in a typical } \boldsymbol{x}\right\} \approx \Pr\left\{x_n = 0\right\}$$

which in our model reads

$$\begin{aligned}\frac{1}{N}\left\{\# \textit{ of zeros in a typical } \boldsymbol{x}\right\} &\approx \Pr\left\{x_n = 0\right\} \\ &= \Pr\left\{b_n = 0\right\} = 1 - \alpha\end{aligned}$$

This approximation gets more precise as N grows very large

Stochastic Model for Sparse Signals

Why is it a good model for sparse signals?

Now let us calculate the ℓ_0-norm of $\boldsymbol{x}$

$$\begin{aligned}\|\boldsymbol{x}\|_0 &= \{\# \text{ of non-zeros in } \boldsymbol{x}\} \\ &= N - \{\# \text{ of zeros in } \boldsymbol{x}\} \\ &\approx N - N \Pr\{x_n = 0\} = N - N(1-\alpha) = N\alpha\end{aligned}$$

So, $\boldsymbol{x}$ is approximately an $N\alpha$-sparse signal

Since α denotes the fraction of non-zero entries

α is often called the sparsity factor

Stochastic Model for Sparse Signals

Why is it a good model for sparse signals?

So this model has two key properties

- *It fits our prior belief that the signal is sparse*
- *The sparsity of the signal is not deterministic*

Why we do not like deterministic sparsity?

Because for practical sparse signals in typical applications

We do not know the exact sparsity

We often only know approximately the sparsity factor

Stochastic Compressive Sensing

Section 2: Optimal Stochastic Compressive Sensing

Optimal Sampling and Recovery

Given the stochastic model, we now answer a fundamental question

What is the optimal approach for compressive sensing?

But haven't we said ℓ_0-norm minimization is optimal?

Well! That was optimal when

We do the sampling linearly via a sampling matrix

Moreover, it does not say

What exactly is the minimum number of samples?

Optimal Sampling and Recovery

Let's make this question clear

We can look at compressive sensing as a system in which

- *Sparse signal $\boldsymbol{x} \in \mathbb{R}^N$ is given to a sampling function*
- *Sampling function compresses $\boldsymbol{x}$ into M samples as*

$$\boldsymbol{y} = f_N(\boldsymbol{x}) \in \mathbb{R}^M$$

- *Samples are then given to a recovery algorithm $g_N(\cdot)$*

$$\hat{\boldsymbol{x}} = g_N(\boldsymbol{y}) \in \mathbb{R}^N$$

Optimal Sampling and Recovery

One could think of this system as below

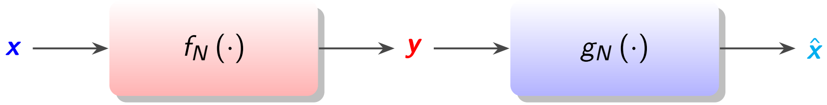
$$\begin{tikzcd} \boldsymbol{x} \arrow[r] & \boxed{f_N(\cdot)} \arrow[r] & \boldsymbol{y} \arrow[r] & \boxed{g_N(\cdot)} \arrow[r] & \hat{\boldsymbol{x}} \end{tikzcd}$$

One might wonder why we write $f_N(\cdot)$ and $g_N(\cdot)$?

$$\boxed{\begin{array}{l} \textit{Well! In the best case, for signal with different lengths} \\ \\ \textit{We can construct different samplers and recovery algorithms} \end{array}}$$

Optimal Sampling and Recovery

In this system



$$\begin{tikzcd} \boldsymbol{x} \arrow[r] & \boxed{f_N(\cdot)} \arrow[r] & \boldsymbol{y} \arrow[r] & \boxed{g_N(\cdot)} \arrow[r] & \hat{\boldsymbol{x}} \end{tikzcd}$$

We are happy, if

1. *We collect minimum number of samples, and*
2. *We have $\hat{\boldsymbol{x}} = \boldsymbol{x}$*

How can we formulate these purposes in this general model?

Optimal Sampling and Recovery

We do it, as we used to do in information theory

Let's define for this system a compression rate

$$R_N = \frac{M}{N}$$

This still depends on N!

Example: *The sampler $f_N(\cdot)$ collects*

- $M = N/2$ *samples if* N *is even*
- $M = (N-1)/2$ *samples if* N *is odd*

Optimal Sampling and Recovery

We do it, as we used to do in information theory

Let's define for this system a compression rate

$$R_N = \frac{M}{N}$$

This still depends on N!

Example: *Clearly, in this case we have*

- $R_N = 0.5$ *for even* N
- $R_N = (N-1)/2N < 0.5$ *for odd* N

Optimal Sampling and Recovery

Also, let's formulate the reliability of our recovery

For this, we define the error probability which is

$$P_N^{\mathrm{e}} = \Pr\left\{\hat{\boldsymbol{x}} \neq \boldsymbol{x}\right\}$$

It again clearly depends on N

We are now interested in the asymptotic case, i.e.,

when $N \to \infty$

Why we look at this case?

Simply because it was the most extreme case

Optimal Sampling and Recovery

Achievable Compression Rate

The rate R is achievable, if there exist sampler $f_N(\cdot)$ and recovery algorithm $g_N(\cdot)$ with the rate

$$\lim_{N\to\infty} R_N = R$$

such that

$$\lim_{N\to\infty} P_N^{\mathrm{e}} = \lim_{N\to\infty} \Pr\left\{ g_N\left(f_N\left(\boldsymbol{x}\right)\right) \neq \boldsymbol{x} \right\} = 0$$

Optimal Sampling and Recovery

Simply speaking, an achievable compression rate is

A rate at which we have perfect recovery

A simple example is $R = 1$

> *In this case, we always have* $M = N$ *and could set*
>
> $$f_N(\boldsymbol{x}) = g_N(\boldsymbol{x}) = \mathbf{I}_N \boldsymbol{x}$$

Clearly

$$\lim_{N\to\infty} R_N = \lim_{N\to\infty} \frac{M}{N} = \lim_{N\to\infty} 1 = 1$$

Optimal Sampling and Recovery

Simply speaking, an achievable compression rate is

A rate at which we can have perfect recovery

A simple example is $R = 1$

In this case, we always have $M = N$ *and could set*

$$f_N(\boldsymbol{x}) = g_N(\boldsymbol{x}) = \mathbf{I}_N \boldsymbol{x}$$

And also we have

$$\lim_{N\to\infty} P_N^{\mathrm{e}} = \lim_{N\to\infty} \Pr\left\{\mathbf{I}_N\left(\mathbf{I}_N \boldsymbol{x}\right) \neq \boldsymbol{x}\right\} = \lim_{N\to\infty} \Pr\left\{\boldsymbol{x} \neq \boldsymbol{x}\right\} = 0$$

Optimal Sampling and Recovery

Now the main question is that given all these degrees of freedom

What is the minimum achievable rate?

This means

What is the minimum compression rate, by which

- *We still can recover the sparse signal from its samples*
- *If we go below, we cannot do perfect recovery anymore*

Optimal Sampling and Recovery

Minimum Prossible Compression Rate

For the given stochastic model, the minimum achievable rate is

$$R^\star = \alpha$$

This rate is achieved by a linear sampler

What does this result say?

- *With large dimensions, we need* $M \approx \alpha N = s$ *samples*
- *Linear sampling is good, we don't need nonlinear samplers*

Optimal Sampling and Recovery: Few Notes

- *The final result is more generally given in terms of*

  Rényi information dimension of the signal

- *For the simple model we considered*

  Rényi information dimension of the signal $= \alpha$

- *This result only talks about minimum compression rate*
- *It doesn't specify*
  - *optimal* $\mathbf{A}$
  - *optimal recovery algorithm*

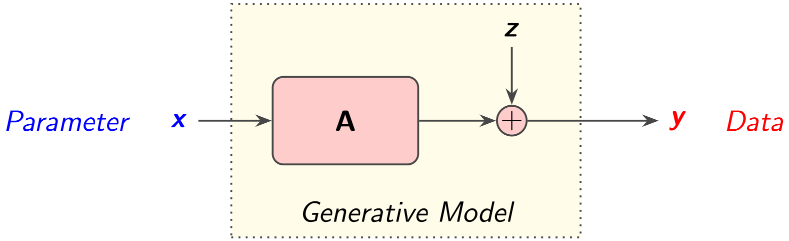
Bayesian Sparse Recovery

Sparse Recovery via Bayesian Inference

We now know that

linear sampling is optimal

Let us look back again at the problem of sparse recovery



Sparse Recovery via Bayesian Inference

We can look at the sparse recovery problem as a

Bayesian inference problem

This means

We want to talk about the parameter $\boldsymbol{x}$ given that

- *we observe data $\boldsymbol{y}$*
- *we know the generative model $\boldsymbol{y} = \mathbf{A}\boldsymbol{x} + \boldsymbol{z}$*

What should we do then?

We should determine the posterior $P(\boldsymbol{x}|\boldsymbol{y})$

Components of the Inference Problem

Elements of the Bayesian inference problem

- *Likelihood of the data given a particular parameter* $\boldsymbol{x}$

$$P(\boldsymbol{y}|\boldsymbol{x}, \mathbf{A}) = P_{\boldsymbol{z}}(\boldsymbol{y} - \mathbf{A}\boldsymbol{x})$$

- *Prior belief on the parameter*

$$P(\boldsymbol{x}) \rightsquigarrow \text{ A probabilistic model for a sparse signal}$$

How do we calculate the posterior?

We use the Bayes rule

Posterior Distribution

We use the Bayes rule

$$P(\boldsymbol{x}|\boldsymbol{y}) = \frac{P(\boldsymbol{y}|\boldsymbol{x},\mathbf{A})\,P(\boldsymbol{x})}{\int P(\boldsymbol{y}|\boldsymbol{x},\mathbf{A})\,P(\boldsymbol{x})\mathrm{d}\boldsymbol{x}} = \frac{P_{\boldsymbol{z}}(\boldsymbol{y}-\mathbf{A}\boldsymbol{x})\,P(\boldsymbol{x})}{\int P_{\boldsymbol{z}}(\boldsymbol{y}-\mathbf{A}\boldsymbol{x})\,P(\boldsymbol{x})\mathrm{d}\boldsymbol{x}}$$

Let's assume a classical distribution for noise

$\boldsymbol{z}$ is i.i.d. Gaussian with mean zero and variance λ

$$P_{\boldsymbol{z}}(\boldsymbol{z}) = \left(\frac{1}{\sqrt{2\pi\lambda}}\right)^{M} \exp\left\{-\frac{\|\boldsymbol{z}\|^2}{2\lambda}\right\}$$

Posterior Distribution

We use the Bayes rule

$$P(\boldsymbol{x}|\boldsymbol{y}) = \frac{\exp\left\{-\frac{\|\boldsymbol{y}-\mathbf{A}\boldsymbol{x}\|^2}{2\lambda}\right\} P(\boldsymbol{x})}{\int \exp\left\{-\frac{\|\boldsymbol{y}-\mathbf{A}\boldsymbol{x}\|^2}{2\lambda}\right\} P(\boldsymbol{x}) \mathrm{d}\boldsymbol{x}}$$

$$= \frac{\exp\left\{-\frac{\|\boldsymbol{y}-\mathbf{A}\boldsymbol{x}\|^2}{2\lambda}\right\} \exp\left\{-\log\frac{1}{P(\boldsymbol{x})}\right\}}{\int \exp\left\{-\frac{\|\boldsymbol{y}-\mathbf{A}\boldsymbol{x}\|^2}{2\lambda}\right\} \exp\left\{-\log\frac{1}{P(\boldsymbol{x})}\right\} \mathrm{d}\boldsymbol{x}}$$

Posterior Distribution

We can hence conclude that the posterior reads

$$P(\boldsymbol{x}|\boldsymbol{y}) = \frac{\exp\{-\mathcal{E}(\boldsymbol{x}|\boldsymbol{y})\}}{\int \exp\{-\mathcal{E}(\boldsymbol{x}|\boldsymbol{y})\} \mathrm{d}\boldsymbol{x}}$$

where the exponent function $\mathcal{E}(\boldsymbol{x}|\boldsymbol{y})$ *is*

$$\mathcal{E}(\boldsymbol{x}|\boldsymbol{y}) = \frac{1}{2\lambda}\|\boldsymbol{y} - \mathbf{A}\boldsymbol{x}\|^2 + \log\frac{1}{P(\boldsymbol{x})}$$

Performing Bayesian Inference

Since we have the posterior,

we can perform Bayesian inference

But, how exactly can we do Bayesian inference?

There are two major approaches

1. *Inferring via maximum-a-posteriori (MAP)*
2. *Inferring via risk minimization*

We now go through these approaches

Performing Bayesian Inference

But, before we start let's get back to posterior's exponent

$$\mathcal{E}(\boldsymbol{x}|\boldsymbol{y}) = \frac{1}{2\lambda}\|\boldsymbol{y} - \mathbf{A}\boldsymbol{x}\|^2 + \log\frac{1}{P(\boldsymbol{x})}$$

Which parameters are postulated by our stochastic model here?

1. *Noise variance* λ
2. *Prior distribution* $P(\boldsymbol{x})$

Keep this fact in mind for later!

Bayesian Sparse Recovery

Section 1: Bayesian Sparse Recovery via MAP

MAP Estimation

MAP estimates the parameter by finding maximal posterior

$$\begin{aligned}\hat{\boldsymbol{x}} &= \operatorname*{argmax}_{\boldsymbol{x}} P\left(\boldsymbol{x}|\boldsymbol{y}\right)\\ &= \operatorname*{argmax}_{\boldsymbol{x}} \frac{\exp\left\{-\mathcal{E}\left(\boldsymbol{x}|\boldsymbol{y}\right)\right\}}{\displaystyle\int \exp\left\{-\mathcal{E}\left(\boldsymbol{x}|\boldsymbol{y}\right)\right\} \mathrm{d}\boldsymbol{x}}\\ &= \operatorname*{argmin}_{\boldsymbol{x}} \mathcal{E}\left(\boldsymbol{x}|\boldsymbol{y}\right)\\ &= \operatorname*{argmin}_{\boldsymbol{x}} \frac{1}{2\lambda}\left\|\boldsymbol{y} - \mathbf{A}\boldsymbol{x}\right\|^2 + \log \frac{1}{P\left(\boldsymbol{x}\right)}\end{aligned}$$

MAP Estimation

MAP Estimation

$$\hat{\boldsymbol{x}} = \operatorname*{argmin}_{\boldsymbol{x}} \frac{1}{2\lambda} \| \boldsymbol{y} - \mathbf{A}\boldsymbol{x} \|^2 + \log \frac{1}{P(\boldsymbol{x})}$$

This recovers the recovery algorithms we already learned

Example: *Let* $P(\boldsymbol{x}) = C \exp\{-\|\boldsymbol{x}\|_1\}$*; then,*

$$\hat{\boldsymbol{x}} = \operatorname*{argmin}_{\boldsymbol{x}} \frac{1}{2\lambda} \| \boldsymbol{y} - \mathbf{A}\boldsymbol{x} \|^2 + \|\boldsymbol{x}\|_1$$

Which is the Basis Pursuit Denoising algorithm

MAP Estimation

MAP Estimation

$$\hat{\boldsymbol{x}} = \operatorname*{argmin}_{\boldsymbol{x}} \frac{1}{2\lambda} \|\boldsymbol{y} - \mathbf{A}\boldsymbol{x}\|^2 + \log \frac{1}{P(\boldsymbol{x})}$$

This recovers the recovery algorithms we already learned

Basis Pursuit Denoising algorithm is a MAP estimator

- *Prior is assumed to be Laplace*
- *λ is the postulated noise variance*

MAP Estimation

MAP Estimation

$$\hat{\boldsymbol{x}} = \operatorname*{argmin}_{\boldsymbol{x}} \frac{1}{2\lambda} \|\boldsymbol{y} - \mathbf{A}\boldsymbol{x}\|^2 + \log \frac{1}{P(\boldsymbol{x})}$$

What happens if we send $\lambda \to 0$

We recover the noiseless case, where

- *We should have* $\boldsymbol{y} = \mathbf{A}\hat{\boldsymbol{x}}$
- *Prior should be proportional to* ℓ_0*-norm*

Bayesian Sparse Recovery

Section 2: Bayesian Sparse Recovery via Risk Minimization

Risk Minimization

Why should MAP estimation be any good?

Well, it finds the estimate which based on the data has

the most posterior likelihood!

But, this is intuition! Is there any math behind it?

Yes! We can show that MAP estimation

minimizes the error probability

Error Probability Minimization via MAP

We want to find estimate $\hat{\boldsymbol{x}}$ with minimum error probability

$$\begin{aligned}\hat{\boldsymbol{x}} &= \operatorname*{argmin}_{\boldsymbol{u}} \Pr\left(\boldsymbol{x} \neq \boldsymbol{u} | \boldsymbol{y}\right)\\ &= \operatorname*{argmin}_{\boldsymbol{u}} 1 - \Pr\left(\boldsymbol{x} = \boldsymbol{u} | \boldsymbol{y}\right)\\ &= \operatorname*{argmax}_{\boldsymbol{u}} \Pr\left(\boldsymbol{x} = \boldsymbol{u} | \boldsymbol{y}\right)\end{aligned}$$

*If we assume that **x** is a signal with discrete components, we have*

$$\Pr\left(\boldsymbol{x} = \boldsymbol{u} | \boldsymbol{y}\right) = P\left(\boldsymbol{x} = \boldsymbol{u} | \boldsymbol{y}\right)$$

which concludes that $\hat{\boldsymbol{x}}$ is the MAP estimate!

Error Probability Minimization via MAP

But, what if the signal has real-valued components?

With noisy observations and real components, we always have

$$\Pr\left(\boldsymbol{x} = \boldsymbol{u} | \boldsymbol{y}\right) = 0$$

for any $\boldsymbol{u}$

This means that

The error probability is always one; and hence,

Error probability minimization does not lead to any estimate!

MAP as Risk Minimization

Let's define the following risk function

$$d(\boldsymbol{x}; \boldsymbol{u}) = 1 - \delta(\boldsymbol{x} - \boldsymbol{u})$$

This function calculates the risk or distortion, when

we estimate $\boldsymbol{x}$ by $\boldsymbol{u}$

As we do not know $\boldsymbol{x}$, we cannot calculate it!

But, we can determine its average based on our data

We should take the mean with respect to the posterior

MAP as Risk Minimization

Let's calculate the average risk for $\boldsymbol{u}$ conditioned to our data $\boldsymbol{y}$

$$\begin{aligned} D(\boldsymbol{u}|\boldsymbol{y}) &= \mathcal{E}\left[d(\boldsymbol{x};\boldsymbol{u})|\boldsymbol{y}\right] \\ &= \int \left[1-\delta(\boldsymbol{x}-\boldsymbol{u})\right] P(\boldsymbol{x}|\boldsymbol{y})\mathrm{d}\boldsymbol{x} \\ &= 1 - P(\boldsymbol{u}|\boldsymbol{y}) \end{aligned}$$

So, we could say that the MAP estimate $\hat{\boldsymbol{x}}$ is

$$\hat{\boldsymbol{x}} = \operatorname*{argmax}_{\boldsymbol{u}} P(\boldsymbol{u}|\boldsymbol{y}) = \operatorname*{argmin}_{\boldsymbol{u}} D(\boldsymbol{u}|\boldsymbol{y})$$

MAP is risk minimization for that given risk function

Risk Minimization

Back to the question: Why should MAP estimation be any good?

Well, it finds the estimate which based on the data has

the most posterior likelihood!

But, this is intuition! Is there any math behind it?

Well! It does risk minimization

Why should risk function be restricted to the choice in MAP?

Well! It actually shouldn't

Bayesian Estimation by Risk Minimization

We can calculate the average risk with respect to any risk function

$$\begin{aligned} D\left(\boldsymbol{u}|\boldsymbol{y}\right) &= \mathcal{E}\left[d\left(\boldsymbol{x};\boldsymbol{u}\right)|\boldsymbol{y}\right] \\ &= \int d\left(\boldsymbol{x};\boldsymbol{u}\right) P\left(\boldsymbol{x}|\boldsymbol{y}\right)\mathrm{d}\boldsymbol{x} \end{aligned}$$

and find the estimate by minimizing it

$$\hat{\boldsymbol{x}} = \operatorname*{argmin}_{\boldsymbol{u}} D\left(\boldsymbol{u}|\boldsymbol{y}\right)$$

Bayesian Estimation by Risk Minimization

What are the common choice for risk function?

The most common is the squared error

$$d\left(\boldsymbol{x};\boldsymbol{u}\right)=\left\|\boldsymbol{x}-\boldsymbol{u}\right\|^2$$

Risk minimization in this case is called

minimum mean squared error (MMSE)

Bayesian Estimation by Risk Minimization

But, we could also use other risk functions

1. *If we use the logarithmic distance between two distributions*

$$d\left(\boldsymbol{x};\boldsymbol{u}\right)=\log\frac{P\left(\boldsymbol{x}|\boldsymbol{y}\right)}{Q_{\boldsymbol{u}}\left(\boldsymbol{x}|\boldsymbol{y}\right)}$$

   we end up with the KL-divergence

2. *We can set risk to the information content of a distribution*

$$d\left(\boldsymbol{x};\boldsymbol{u}\right)=\log\frac{1}{Q_{\boldsymbol{u}}\left(\boldsymbol{x}|\boldsymbol{y}\right)}$$

   which ends up with the cross entropy

MMSE Estimation

Let's focus on the classical case

In MMSE, we minimize the average of the following risk

$$d\left(\boldsymbol{x};\boldsymbol{u}\right)=\left\|\boldsymbol{x}-\boldsymbol{u}\right\|^2$$

We find the minimizer of

$$D\left(\boldsymbol{u}|\boldsymbol{y}\right)=\mathcal{E}\left[\left\|\boldsymbol{x}-\boldsymbol{u}\right\|^2|\boldsymbol{y}\right]$$

Since we optimize a convex average, at optimum $\boldsymbol{u}=\hat{\boldsymbol{x}}$ *we have*

$$\nabla_{\boldsymbol{u}} D\left(\boldsymbol{u}|\boldsymbol{y}\right)|_{\boldsymbol{u}=\hat{\boldsymbol{x}}}=\mathbf{0}$$

MMSE Estimation

Let's focus on the following on the classical case

In MMSE, we minimize the average of the following risk

$$d\left(\boldsymbol{x}; \boldsymbol{u}\right) = \|\boldsymbol{x} - \boldsymbol{u}\|^2$$

As linear operators exchange, we have

$$\begin{aligned}\nabla_{\boldsymbol{u}} D\left(\boldsymbol{u}|\boldsymbol{y}\right) &= \mathcal{E}\left[\nabla_{\boldsymbol{u}}\|\boldsymbol{x}-\boldsymbol{u}\|^2|\boldsymbol{y}\right]\\ &= -2\mathcal{E}\left[\boldsymbol{x}-\boldsymbol{u}|\boldsymbol{y}\right]\\ &= -2\left(\mathcal{E}\left[\boldsymbol{x}|\boldsymbol{y}\right]-\mathcal{E}\left[\boldsymbol{u}|\boldsymbol{y}\right]\right)\\ &= -2\left(\mathcal{E}\left[\boldsymbol{x}|\boldsymbol{y}\right]-\boldsymbol{u}\right)\end{aligned}$$

MMSE Estimation

Let's focus on the following on the classical case

In MMSE, we minimize the average of the following risk

$$d\left(\boldsymbol{x};\boldsymbol{u}\right)=\left\|\boldsymbol{x}-\boldsymbol{u}\right\|^2$$

Since we have at the MMSE point $\boldsymbol{u}=\hat{\boldsymbol{x}}$

$$\nabla_{\boldsymbol{u}} D\left(\boldsymbol{u}|\boldsymbol{y}\right)|_{\boldsymbol{u}=\hat{\boldsymbol{x}}}=\boldsymbol{0}$$

we can conclude that

$$\hat{\boldsymbol{x}}=\mathcal{E}\left[\boldsymbol{x}|\boldsymbol{y}\right]=\int \boldsymbol{x}P\left(\boldsymbol{x}|\boldsymbol{y}\right)\mathrm{d}\boldsymbol{x}$$

MMSE Estimation

MMSE Estimator

MMSE is given by average the stochastic signal via the posterior

$$\hat{\boldsymbol{x}} = \mathcal{E}\left[\boldsymbol{x} | \boldsymbol{y}\right] = \int \boldsymbol{x} P\left(\boldsymbol{x} | \boldsymbol{y}\right) \mathrm{d}\boldsymbol{x}$$

1. *MMSE estimator is not necessarily linear! Hence,*
   MMSE estimator is different from LMMSE estimator!
2. *With Gaussian prior, MMSE and LMMSE are the same*

Introduction to AMP

Calculating MMSE Estimate

AMP stands for

Approximate Message Passing

AMP often refers to an iterative algorithm which

$$\boxed{\textit{implements a Bayesian technique with a feasible complexity}}$$

An AMP algorithm . . .

- *starts from message passing to compute Bayesian recovery*
- *uses approximations to deal with loops in the factor-graph*

We now learn basics of AMP

Introduction to AMP

Section 1: Complexity of Bayesian Estimation

Calculating MMSE Estimate

To calculate the MMSE estimate, we should calculate

$$\hat{x}_n = \int x_n P\left(\boldsymbol{x} | \boldsymbol{y}\right) \mathrm{d}\boldsymbol{x}$$

for every $n \in [N]$

Let's back to the original form of the posterior

$$P\left(\boldsymbol{x} | \boldsymbol{y}\right) = \frac{\exp\left\{-\frac{\|\boldsymbol{y} - \mathbf{A}\boldsymbol{x}\|^2}{2\lambda}\right\} P\left(\boldsymbol{x}\right)}{\int \exp\left\{-\frac{\|\boldsymbol{y} - \mathbf{A}\boldsymbol{x}\|^2}{2\lambda}\right\} P\left(\boldsymbol{x}\right) \mathrm{d}\boldsymbol{x}}$$

Calculating MMSE Estimate

To calculate the MMSE estimate, we should calculate

$$\hat{x}_n = \int x_n P(\boldsymbol{x}|\boldsymbol{y}) \mathrm{d}\boldsymbol{x}$$

for every $n \in [N]$

Therefore, the integral is of the form

$$\hat{x}_n = \frac{\displaystyle\int x_n \exp\left\{-\frac{\|\boldsymbol{y} - \mathbf{A}\boldsymbol{x}\|^2}{2\lambda}\right\} P(\boldsymbol{x}) \mathrm{d}\boldsymbol{x}}{\displaystyle\int \exp\left\{-\frac{\|\boldsymbol{y} - \mathbf{A}\boldsymbol{x}\|^2}{2\lambda}\right\} P(\boldsymbol{x}) \mathrm{d}\boldsymbol{x}}$$

Calculating MMSE Estimate

In fact, it is computationally enough if we can calculate

$$\mathcal{Z} = \int \exp\left\{-\frac{\|\boldsymbol{y} - \mathbf{A}\boldsymbol{x}\|^2}{2\lambda}\right\} P(\boldsymbol{x}) \mathrm{d}\boldsymbol{x}$$

What does it mean?

It means that

If my computer can determine $\mathcal{Z}$ $\rightsquigarrow$ It can easily calculate $\hat{x}_n$

So, the main question is how complex is it to calculate $\mathcal{Z}$?

In statistical physics, $\mathcal{Z}$ is called the partition function of posterior

Calculating MMSE Estimate

You may wonder why?!

Well, it's coming from averaging trick in statistical physics: Let

$$I(h) = \int \exp\left\{ -\frac{\|\boldsymbol{y} - \mathbf{A}\boldsymbol{x}\|^2}{2\lambda} + h x_n \right\} P(\boldsymbol{x}) \mathrm{d}\boldsymbol{x}$$

Then, we have

$$\hat{x}_n = \frac{1}{\mathcal{Z}} \frac{\partial}{\partial h} I(h) \big|_{h=0}$$

If our computer can calculate $\mathcal{Z} \rightsquigarrow$ *It can for sure calculate* $I(h)$

Complexity of MMSE Estimator

We need to calculate

$$\mathcal{Z} = \int \exp\left\{-\frac{\|\boldsymbol{y} - \mathbf{A}\boldsymbol{x}\|^2}{2\lambda}\right\} P(\boldsymbol{x}) \mathrm{d}\boldsymbol{x}$$

Let's consider the classical i.i.d. sparse prior

$$P(\boldsymbol{x}) = \prod_{n=1}^{N} P(x_n)$$

x_n is product of an α-Bernoulli and a real random variable; thus

$$P(x_n) = (1-\alpha)\,\delta(x_n) + \alpha Q(x_n)$$

Complexity of MMSE Estimator

Starting with calculation, we have

$$\mathcal{Z} = \int \exp\left\{-\frac{\|\boldsymbol{y} - \mathbf{A}\boldsymbol{x}\|^2}{2\lambda}\right\} \prod_{n=1}^{N} P(x_n)\mathrm{d}x_n$$

$$= \int \underbrace{\left[\int \exp\left\{-\frac{\|\boldsymbol{y} - \mathbf{A}\boldsymbol{x}\|^2}{2\lambda}\right\} P(x_1)\mathrm{d}x_1\right]}_{\mathcal{Z}_1\left(\boldsymbol{x}_{\sim[1]}\right)} \prod_{n=2}^{N} P(x_n)\mathrm{d}x_n$$

$\boldsymbol{x}_{\sim[n]}$ *means* $\boldsymbol{x}$ *without* $x_1, \dots, x_n$, *i.e.*, $\boldsymbol{x}_{\sim[n]} = [x_{n+1}, \dots, x_N]^\mathsf{T}$

Let's now focus on the first integral

Complexity of MMSE Estimator

We first define some expressions:

Reduced difference $\boldsymbol{v}_n$ *as*

$$\boldsymbol{v}_n = \boldsymbol{y} - \sum_{i=n+1}^{N} x_i \boldsymbol{a}_i$$

with $\boldsymbol{a}_n$ *being n-th column on* $\mathbf{A}$

1. $\boldsymbol{v}_0 = \boldsymbol{y} - \mathbf{A}\boldsymbol{x}$
2. $\boldsymbol{v}_n$ *is a function of* $\boldsymbol{x}_{\sim[n]}$
3. $\boldsymbol{v}_{n-1} = \boldsymbol{v}_n - x_n \boldsymbol{a}_n$

Complexity of MMSE Estimator

We first define some expressions:

Sparsity term $\mathcal{K}(\boldsymbol{v})$ *as*

$$\mathcal{K}(\boldsymbol{v}) = \exp\left\{-\frac{\|\boldsymbol{v}\|^2}{2\lambda}\right\}$$

Complexity of MMSE Estimator

We first define some expressions:

Exponential average $\mathcal{F}_0(\boldsymbol{v};\boldsymbol{a})$ *as*

$$\mathcal{F}_0(\boldsymbol{v};\boldsymbol{a}) = \int \exp\left\{-\frac{\|\boldsymbol{v}-\boldsymbol{a}x\|^2}{2\lambda}\right\} Q(x)\mathrm{d}x$$

and its j*-th order integral as*

$$\mathcal{F}_{j+1}(\boldsymbol{v};\mathbf{A}_{j+1}) = \int \mathcal{F}_j(\boldsymbol{v}-\boldsymbol{a}_{j+1}x;\mathbf{A}_j)\, Q(x)\mathrm{d}x$$

where $\mathbf{A}_{j+1} = [\mathbf{A}_j, \boldsymbol{a}_{j+1}]$

Complexity of MMSE Estimator

Back to the calculation, we have

$$
\begin{aligned}
\mathcal{Z}_1\left(\boldsymbol{x}_{\sim[1]}\right) &= \int \exp\left\{-\frac{\|\boldsymbol{y}-\mathbf{A}\boldsymbol{x}\|^2}{2\lambda}\right\} P\left(x_1\right)\mathrm{d}x_1 \\
&= \int \exp\left\{-\frac{\|\boldsymbol{v}_0\|^2}{2\lambda}\right\} P\left(x_1\right)\mathrm{d}x_1 \\
&= \int \exp\left\{-\frac{\|\boldsymbol{v}_1 - x_1\boldsymbol{a}_1\|^2}{2\lambda}\right\} \left[\left(1-\alpha\right)\delta\left(x_1\right)+\alpha Q\left(x_1\right)\right]\mathrm{d}x_1 \\
&= \left(1-\alpha\right)\exp\left\{-\frac{\|\boldsymbol{v}_1\|^2}{2\lambda}\right\} \equiv \left(1-\alpha\right)\mathcal{K}\left(\boldsymbol{v}_1\right) \\
&\quad + \alpha \int \exp\left\{-\frac{\|\boldsymbol{v}_1 - x_1\boldsymbol{a}_1\|^2}{2\lambda}\right\} Q\left(x_1\right)\mathrm{d}x_1 \equiv \alpha\mathcal{F}_0\left(\boldsymbol{v}_1;\boldsymbol{a}_1\right)
\end{aligned}
$$

Complexity of MMSE Estimator

So, we have concluded that

$$\mathcal{Z}_1\left(\boldsymbol{x}_{\sim[1]}\right) = (1-\alpha)\mathcal{K}\left(\boldsymbol{v}_1\right) + \alpha\mathcal{F}_0\left(\boldsymbol{v}_1; \boldsymbol{a}_1\right)$$

How many terms should be added at this step?

At this point 2 terms

Let's continue!

Complexity of MMSE Estimator

Back to calculation of $\mathcal{Z}$*, we have*

$$\mathcal{Z} = \int \mathcal{Z}_1\left(\boldsymbol{x}_{\sim[1]}\right)\prod_{n=2}^{N} P\left(x_n\right)\mathrm{d}x_n$$
$$= \int \underbrace{\left[\int \mathcal{Z}_1\left(\boldsymbol{x}_{\sim[1]}\right) P\left(x_2\right)\mathrm{d}x_2\right]}_{\mathcal{Z}_2\left(\boldsymbol{x}_{\sim[2]}\right)}\prod_{n=3}^{N} P\left(x_n\right)\mathrm{d}x_n$$

Let's go for $\mathcal{Z}_2\left(\boldsymbol{x}_{\sim[2]}\right)$

Complexity of MMSE Estimator

Using our earlier calculations, we have

$$\mathcal{Z}_2\left(\boldsymbol{x}_{\sim[2]}\right) = \int \left[(1-\alpha)\mathcal{K}\left(\boldsymbol{v}_1\right) + \alpha\mathcal{F}_0\left(\boldsymbol{v}_1;\boldsymbol{a}_1\right)\right] P\left(x_2\right)\mathrm{d}x_2$$
$$= \int \left[(1-\alpha)\mathcal{K}\left(\boldsymbol{v}_2 - x_2\boldsymbol{a}_2\right) + \alpha\mathcal{F}_0\left(\boldsymbol{v}_2 - x_2\boldsymbol{a}_2;\boldsymbol{a}_1\right)\right] P\left(x_2\right)\mathrm{d}x_2$$

If we use our defined expressions, we end up with

$$\mathcal{Z}_2\left(\boldsymbol{x}_{\sim[2]}\right) = \beta_1\mathcal{K}\left(\boldsymbol{v}_1\right) + \beta_2\mathcal{F}_0\left(\boldsymbol{v}_2;\boldsymbol{a}_1\right) + \beta_2\mathcal{F}_0\left(\boldsymbol{v}_2;\boldsymbol{a}_2\right) + \beta_3\mathcal{F}_1\left(\boldsymbol{v}_2;\mathbf{A}_2\right)$$

where $\mathbf{A}_2 = [\boldsymbol{a}_1, \boldsymbol{a}_2]$ *and*

$$\beta_1 = (1-\alpha)^2 \qquad \beta_2 = \alpha(1-\alpha) \qquad \beta_3 = \alpha^2$$

Complexity of MMSE Estimator

Similar to $\mathcal{Z}_1\left(\boldsymbol{x}_{\sim[1]}\right)$, *the coefficients add up to one, i.e.,*

$$\beta_1 + 2\beta_2 + \beta_3 = 1$$

Given the expression for $\mathcal{Z}_2\left(\boldsymbol{x}_{\sim[2]}\right)$ ...

How many terms should be added at this step?

At this point, $2^2 = 4$ *terms*

Complexity of MMSE Estimator

To finish calculation, we should continue repeating this procedure

1. *In step* $n+1$, *we get*

$$\mathcal{Z}_{n+1}\left(\boldsymbol{x}_{\sim[n+1]}\right) = (1-\alpha)\mathcal{Z}_n\left(\boldsymbol{x}_{\sim[n]}\right)|_{x_{n+1}=0} + \alpha\int \mathcal{Z}_n\left(\boldsymbol{x}_{\sim[n]}\right) Q\left(x_n\right)\mathrm{d}x_n$$

2. $\mathcal{Z}_n\left(\boldsymbol{x}_{\sim[n]}\right)$ *contains* 2^n *terms*

We finish at step N, *since* $\mathcal{Z}_N\left(\boldsymbol{x}_{\sim[N]}\right) \equiv \mathcal{Z}$

We can hence conclude that in total we add 2^N *terms!*

MMSE estimator has exponential computational complexity

Introduction to AMP

Section 2: Approximate Message Passing

Going on Factor Graphs

Remember the MMSE estimator

$$\hat{x}_n = \frac{\displaystyle\int x_n \exp\left\{-\frac{\|\boldsymbol{y} - \mathbf{A}\boldsymbol{x}\|^2}{2\lambda}\right\} P(\boldsymbol{x}) \mathrm{d}\boldsymbol{x}}{\displaystyle\int \exp\left\{-\frac{\|\boldsymbol{y} - \mathbf{A}\boldsymbol{x}\|^2}{2\lambda}\right\} P(\boldsymbol{x}) \mathrm{d}\boldsymbol{x}}$$

MMSE estimator determines a marginalization which we can do via

Sum-Product Algorithm

Going on Factor Graphs

To see this, let's define the following factors

The linear mixture terms

$$H_m(\boldsymbol{x}) = \exp\left\{ -\frac{1}{2\lambda} \left(y_m - \boldsymbol{b}_m^{\mathsf{T}} \boldsymbol{x} \right)^2 \right\}$$

with $\boldsymbol{b}_m^{\mathsf{T}}$ *being the m-th row of* $\mathbf{A}$

Clearly we have

$$\exp\left\{ -\frac{\|\boldsymbol{y} - \mathbf{A}\boldsymbol{x}\|^2}{2\lambda} \right\} = \prod_{m=1}^{M} H_m(\boldsymbol{x})$$

Going on Factor Graphs

To see this, let's define the following factors

The prior terms

$$P_n\left(x_n\right) = P\left(x_n\right)$$

Clearly we have

$$P\left(\boldsymbol{x}\right) = \prod_{n=1}^{N} P_n\left(x_n\right)$$

Going on Factor Graphs

Now, we define the factorized function $F(\boldsymbol{x})$

$$F(\boldsymbol{x}) = \prod_{m=1}^{M} H_m(\boldsymbol{x}) \prod_{n=1}^{N} P_n(x_n)$$

From information theory, we remember the following definitions

Marginal Function of x_n

$$Z_n(x_n) = \int F(\boldsymbol{x}) \prod_{i \neq n} \mathrm{d}x_i$$

Going on Factor Graphs

Now, we define the factorized function $F(\boldsymbol{x})$

$$F(\boldsymbol{x}) = \prod_{m=1}^{M} H_m(\boldsymbol{x}) \prod_{n=1}^{N} P_n(x_n)$$

From information theory, we remember the following definitions

Global Marginal

$$Z = \int F(\boldsymbol{x}) \prod_{n=1}^{N} \mathrm{d}x_n$$

Going on Factor Graphs

Now, let's get back to MMSE estimate

$$\hat{x}_n = \frac{\displaystyle\int x_n \exp\left\{-\frac{\|\boldsymbol{y} - \mathbf{A}\boldsymbol{x}\|^2}{2\lambda}\right\} P(\boldsymbol{x}) \mathrm{d}\boldsymbol{x}}{\displaystyle\int \exp\left\{-\frac{\|\boldsymbol{y} - \mathbf{A}\boldsymbol{x}\|^2}{2\lambda}\right\} P(\boldsymbol{x}) \mathrm{d}\boldsymbol{x}}$$

$$= \frac{\displaystyle\int x_n \prod_{m=1}^{M} H_m(\boldsymbol{x}) \prod_{n=1}^{N} P_n(x_n) \mathrm{d}\boldsymbol{x}}{\displaystyle\int \prod_{m=1}^{M} H_m(\boldsymbol{x}) \prod_{n=1}^{N} P_n(x_n) \mathrm{d}\boldsymbol{x}} = \frac{\displaystyle\int x_n F(\boldsymbol{x}) \,\mathrm{d}\boldsymbol{x}}{\displaystyle\int F(\boldsymbol{x}) \,\mathrm{d}\boldsymbol{x}}$$

Going on Factor Graphs

Now, let's get back to MMSE estimate

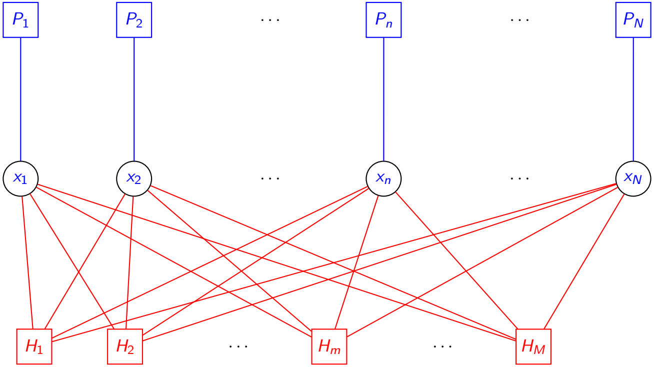
$$\hat{x}_n = \frac{\int x_n \left[\int F(\boldsymbol{x}) \prod_{i \neq n} \mathrm{d}x_i \right] \mathrm{d}x_n}{\int F(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x}} = \frac{\int x_n Z_n(x_n) \, \mathrm{d}x_n}{Z}$$

We know how to calculate them on the factor graph

via sum-product algorithm

Going on Factor Graphs

Let's look at the factor graph



Going on Factor Graphs

Let's write messages from variable nodes to factor nodes

$$q_{x_n \to P_n}(x_n) = \prod_{m=1}^{M} r_{H_m \to x_n}(x_n)$$
$$q_{x_n \to H_m}(x_n) = P_n(x_n) \prod_{j \neq m} r_{H_j \to x_n}(x_n)$$

and messages from factor nodes to variable nodes

$$r_{P_n \to x_n}(x_n) = P_n(x_n)$$
$$r_{H_m \to x_n}(x_n) = \int H_m(\boldsymbol{x}) \prod_{i \neq n} q_{x_i \to H_m}(x_i) \prod_{i \neq n} \mathrm{d}x_i$$

Going on Factor Graphs

We could then find the marginal $Z_n(x_n)$ *as*

$$\begin{aligned} Z_n(x_n) &= r_{P_n \to x_n}(x_n) \prod_{m=1}^{M} r_{H_m \to x_n}(x_n) \\ &= P_n(x_n) \prod_{m=1}^{M} r_{H_m \to x_n}(x_n) \end{aligned}$$

and the global marginal as

$$Z = \int Z_n(x_n)\,\mathrm{d}x_n = \int P_n(x_n) \prod_{m=1}^{M} r_{H_m \to x_n}(x_n)\,\mathrm{d}x_n$$

Deriving AMP

But, can we directly go on the factor graph?

No! The factor graph is super loopy!

So, we may follow the iterative approach in LDPC decoding!

OK! Let's try it

Deriving AMP

First, we note that $\star$ terms are redundant

$$\star \qquad q_{x_n \to P_n}(x_n) = \prod_{m=1}^{M} r_{H_m \to x_n}(x_n)$$

$$q_{x_n \to H_m}(x_n) = P_n(x_n) \prod_{j \neq m} r_{H_j \to x_n}(x_n)$$

and messages from factor nodes to variable nodes

$$\star \qquad r_{P_n \to x_n}(x_n) = P_n(x_n)$$

$$r_{H_m \to x_n}(x_n) = \int H_m(\boldsymbol{x}) \prod_{i \neq n} q_{x_i \to H_m}(x_i) \prod_{i \neq n} \mathrm{d}x_i$$

Deriving AMP

So, we mainly have to deal with

$$r_{H_m \to x_n}(x_n) = \int H_m(\boldsymbol{x}) \prod_{i \neq n} q_{x_i \to H_m}(x_i) \prod_{i \neq n} \mathrm{d}x_i$$

$$q_{x_n \to H_m}(x_n) = P_n(x_n) \prod_{j \neq m} r_{H_j \to x_n}(x_n)$$

We can start with some $q^{(0)}_{x_n \to H_m}(x_n)$ for all m and n and iterate as

$$r^{(t+1)}_{H_m \to x_n}(x_n) = \int H_m(\boldsymbol{x}) \prod_{i \neq n} q^{(t)}_{x_i \to H_m}(x_i) \prod_{i \neq n} \mathrm{d}x_i$$

$$q^{(t+1)}_{x_n \to H_m}(x_n) = P_n(x_n) \prod_{j \neq m} r^{(t+1)}_{H_j \to x_n}(x_n)$$

Deriving AMP

It is however still complex to be calculated, as we have

$$r^{(t+1)}_{H_m \to x_n}(x_n) = \int H_m(\boldsymbol{x}) \prod_{i \neq n} q^{(t)}_{x_i \to H_m}(x_i) \prod_{i \neq n} \mathrm{d}x_i$$

Since the factor graph is dense, this is not an easy integral!

But high density helps us determining a good approximation

This comes intuitively from the central limit theorem

Deriving AMP

Let us open the integral a bit

$$r_{H_m \to x_n}^{(t+1)}(x_n) = \int H_m(\boldsymbol{x}) \prod_{i \neq n} q_{x_i \to H_m}^{(t)}(x_i) \prod_{i \neq n} \mathrm{d}x_i$$
$$= \int \exp\left\{ -\frac{1}{2\lambda} \left(y_m - \boldsymbol{b}_m^\mathsf{T} \boldsymbol{x} \right)^2 \right\} \prod_{i \neq n} q_{x_i \to H_m}^{(t)}(x_i) \prod_{i \neq n} \mathrm{d}x_i$$

We note that

$$y_m - \boldsymbol{b}_m^\mathsf{T} \boldsymbol{x} = y_m - \sum_{n=1}^{N} A_{mn} x_n$$

where $A_{mn} = [\mathbf{A}]_{mn}$

Deriving AMP

Let us open the integral a bit

$$r^{(t+1)}_{H_m \to x_n}(x_n) = \int \exp\left\{ -\frac{1}{2\lambda} \left(y_m - \boldsymbol{b}_m^\mathsf{T} \boldsymbol{x} \right)^2 \right\} \prod_{i \neq n} q^{(t)}_{x_i \to H_m}(x_i) \prod_{i \neq n} \mathrm{d}x_i$$

We can hence say that

$$y_m - \boldsymbol{b}_m^\mathsf{T} \boldsymbol{x} = y_m - \sum_{i \neq n} A_{mi} x_i - A_{mn} x_n = w_{mn} - A_{mn} x_n$$

Let us put it back into the integral

Deriving AMP

Let us open the integral a bit

$$r_{H_m \to x_n}^{(t+1)}(x_n) = \int \exp\left\{ -\frac{\left(w_{mn} - A_{mn} x_n\right)^2}{2\lambda} \right\} \prod_{i \neq n} q_{x_i \to H_m}^{(t)}(x_i) \prod_{i \neq n} \mathrm{d}x_i$$

In this integral we can interpret $q_{x_i \to H_m}^{(t)}(x_i)$ to be

the distribution of x_i in iteration t considered by factor m

1. *These distributions assume x_i's to be independent*
2. *The integral calculates expectation over a function of x_i's*

Deriving AMP

This function only depends on w_{mn}

$$r_{H_m \to x_n}^{(t+1)}\left(x_n\right) = \mathcal{E}\left[\exp\left\{-\frac{\left(w_{mn} - A_{mn}x_n\right)^2}{2\lambda}\right\}\right]$$

So, we can calculate it also as

$$r_{H_m \to x_n}^{(t+1)}\left(x_n\right) = \int \exp\left\{-\frac{\left(w_{mn} - A_{mn}x_n\right)^2}{2\lambda}\right\} \Phi_{mn}^{(t)}\left(w_{mn}\right) \mathrm{d} w_{mn}$$

if we know the distribution $\Phi_{mn}^{(t)}\left(w_{mn}\right)$

Deriving AMP

We approximate $\Phi_{mn}^{(t)}(w_{mn})$ *by a Gaussian distribution*

$$\Phi_{mn}^{(t)}(w_{mn}) \equiv \mathcal{N}\left(\mu_{mn}^{(t)}, \tau^{(t)}\right)$$

Why should it be valid?

There is a mathematical proof behind it that when

M and N are large and the signal and matrix behave well

the limiting distribution is Gaussian

Deriving AMP

We approximate $\Phi_{mn}^{(t)}(w_{mn})$ by a Gaussian distribution

$$\Phi_{mn}^{(t)}(w_{mn}) \equiv \mathcal{N}\left(\mu_{mn}^{(t)}, \tau^{(t)}\right)$$

Intuitively, this is valid as we have

$$w_{mn} = y_m - \sum_{i \neq n} A_{mi} x_i$$

- *A large number of x_i's are adding up*
- *They are independent*

So, central limit theorem suggests it to be correct!

Deriving AMP

We approximate $\Phi_{mn}^{(t)}(w_{mn})$ *by a Gaussian distribution*

$$\Phi_{mn}^{(t)}(w_{mn}) \equiv \mathcal{N}\left(\mu_{mn}^{(t)}, \tau^{(t)}\right)$$

Why mean depends on the indices but variance not?

Well, It comes from high concentration of higher moments!

Deriving AMP

Using this Gaussian approximation, we have

$$r_{H_m \to x_n}^{(t+1)}(x_n) = \mathcal{E}\left[\exp\left\{-\frac{(w_{mn} + A_{mn}x_n)^2}{2\lambda}\right\}\right]$$
$$\propto \exp\left\{-\frac{\left(A_{mn}x_n - \mu_{mn}^{(t)}\right)^2}{2\left(\tau^{(t)} + \lambda\right)}\right\}$$

We need still find $\mu_{mn}^{(t)}$ and $\tau^{(t)}$

We talk about it later

Deriving AMP

Using this Gaussian approximation, we conclude that

$$r^{(t+1)}_{H_m \to x_n}(x_n) \propto \exp\left\{ -\frac{\left(A_{mn}x_n - \mu^{(t)}_{mn}\right)^2}{2\left(\tau^{(t)} + \lambda\right)} \right\}$$

which looks like a Gaussian likelihood for x_n

$$\mu^{(t)}_{mn} = A_{mn}x_n + \xi^{(t)}_n$$

where $\xi^{(t)}_n \sim \mathcal{N}\left(0, \tau^{(t)} + \lambda\right)$

Deriving AMP

Let determine the marginal $Z_n(x_n)$ via our approximation

$$Z_n^{(t)}(x_n) = P_n(x_n) \prod_{m=1}^{M} r_{H_m \to x_n}^{(t)}(x_n)$$

$r_{H_m \to x_n}^{(t)}(x_n)$*'s are all Gaussian likelihood for* x_n

$$\prod_{m=1}^{M} r_{H_m \to x_n}^{(t)}(x_n) \propto \exp\left\{ -\frac{1}{2\varsigma^{(t)}} \left(x_n - \theta_n^{(t)} \right)^2 \right\}$$

Deriving AMP

How does the MMSE estimate in iteration t look like then?

$$\hat{x}_n^{(t)} = \frac{\displaystyle\int x_n Z_n^{(t)}\left(x_n\right) \mathrm{d}x_n}{Z^{(t)}} = \frac{\displaystyle\int x_n \exp\left\{-\frac{1}{2\varsigma^{(t)}}\left(x_n - \theta_n^{(t)}\right)^2\right\} P\left(x_n\right)\mathrm{d}x_n}{\displaystyle\int \exp\left\{-\frac{1}{2\varsigma^{(t)}}\left(x_n - \theta_n^{(t)}\right)^2\right\} P\left(x_n\right)\mathrm{d}x_n}$$

Deriving AMP

The interesting observation is that

$\hat{x}_n^{(t)}$ *is in fact the MMSE estimate of* x_n *from*

$$\theta_n^{(t)} = x_n + \zeta^{(t)}$$

where $\zeta_n^{(t)} \sim \mathcal{N}\left(0, \varsigma^{(t)}\right)$

It remains to find $\theta_n^{(t)}$ *and* $\varsigma^{(t)}$ *...*

Well they are easily calculated by

approximating the second update rule

Final Points

Summary

Stochastic analysis has several applications in compressive sensing

- *We could design good sampling matrices*

  simply generate Gaussian matrices

- *It also lets us understand the efficiency of our approaches*
  - *We found out that linear sampling is actually good*
  - *This means that what we studied is efficient*
- *It further allows for Bayesian compressive sensing*
  - *We find out how to do sparse recovery in a Bayesian framework*
  - *It leads to approximate message passing* algorithms

Which Parts of Textbooks?

$\boxed{\textit{We are over with this part and the course}}$

I would suggest to go over

Statistical Mechanics of Regularized Least Squares
A. Bereyhi, 2020

and study Chapter 6. You could also checkout

- *Shannon Theory for Compressed Sensing, Y. Wu, 2011*
- *How to Design Message Passing Algorithms for Compressed Sensing, D. Donoho, et al, 2011*

Vladimir Vapnik

$\boxed{\textit{We are over with this part and the course}}$

Also it is worth mentioning the name

Vladimir Vapnik

who developed principles of statistical learning a long time a go!

Simply check out

An Overview of Statistical Learning Theory, V. Vapnik, 1999

You probably get very impressed!